IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
Before the Board of Patent Appeals and Interferences

In re Patent Application of: WATT
Serial No. 10/714,481
Filed: November 17, 2003
Title: APPARATUS AND METHOD FOR CONTROLLING ACCESS TO A MEMORY UNIT

Atty Dkt. 550-481
C# M#
TC/A.U.: 2189
Examiner: Flournoy, Horace L.
Date: November 16, 2006

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

☐ **Correspondence Address Indication Form Attached.**

☐ **NOTICE OF APPEAL**
Applicant hereby **appeals** to the Board of Patent Appeals and Interferences from the last decision of the Examiner twice/finally rejecting applicant's claim(s). $500.00 (1401)/$250.00 (2401) $

☒ An appeal **BRIEF** is attached in the pending appeal of the above-identified application. **FEE PREVIOUSLY PAID** $500.00 (1402)/$250.00 (2402) $

☐ Credit for fees paid in prior appeal without decision on merits -$ ( )

☒ Response to Notice of Non-Compliant Appeal Brief.

☐ Petition is hereby made to extend the current due date so as to cover the filing date of this paper and attachment(s)
One Month Extension $120.00 (1251)/$60.00 (2251)
Two Month Extensions $450.00 (1252)/$225.00 (2252)
Three Month Extensions $1020.00 (1253/$510.00 (2253)
Four Month Extensions $1590.00 (1254/$795.00 (2254) $

☐ "Small entity" statement attached.

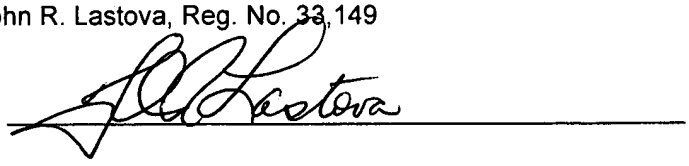
Less month extension previously paid on -$( )

TOTAL FEE ENCLOSED $ 0.00

Any future submission requiring an extension of time is hereby stated to include a petition for such time extension. The Commissioner is hereby authorized to charge any deficiency, or credit any overpayment, in the fee(s) filed, or asserted to be filed, or which should have been filed herewith (or with any paper hereafter filed in this application by this firm) to our **Account No. 14-1140.** A duplicate copy of this sheet is attached.

901 North Glebe Road, 11th Floor
Arlington, Virginia 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100
JRL:maa

NIXON & VANDERHYE P.C.
By Atty: John R. Lastova, Reg. No. 33,149

Signature: 

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

WATT — Atty. Ref.: 550-481

Serial No. 10/714,481 — TC/A.U.: 2189

Filed: November 17, 2003 — Examiner: Flournoy, Horace L.

For: APPARATUS AND METHOD FOR CONTROLLING ACCESS TO A MEMORY UNIT

* * * * * * * * * * * *

November 16, 2006

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

RESPONSE TO NOTICE OF NON-COMPLIANT APPEAL BRIEF

In response to the Notice of Non-Compliant Appeal Brief (mailed November 6, 2006), Applicants hereby submit a new Appeal Brief in which the summary of the invention has been mapped to independent claims 1 and 11, as required in the Notice. In a telephone call with Ms. Lorenda Hood, it was confirmed that the several references to independent claims 1 and 11 throughout the Summary would make the appeal compliant. If there are any further questions, please do not hesitate to contact the undersigned at the telephone number noted below.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: _____
John R. Lastova
Reg. No. 33,149

JRL:maa
901 North Glebe Road, 11th Floor
Arlington, VA 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

WATT

Atty. Ref.: 550-481

Serial No. 10/714,481

Group: 2189

Filed: November 17, 2003

Examiner: Flournoy, Horace L.

For: APPARATUS AND METHOD FOR CONTROLLING ACCESS TO A MEMORY UNIT

Before the Board of Patent Appeals and Interferences

BRIEF FOR APPELLANT

On Appeal From Final Rejection
From Group Art Unit 2189

John R. Lastova
NIXON & VANDERHYE P.C.
11th Floor, 901 North Glebe Road
Arlington, Virginia 22203-1808
(703) 816-4025
Attorney for Appellant
Watt
ARM Limited

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of

WATT — Atty. Ref.: 550-481

Serial No. 10/714,481 — Group: 2189

Filed: November 17, 2003 — Examiner: Flournoy, Horace L.

For: APPARATUS AND METHOD FOR CONTROLLING ACCESS TO A MEMORY UNIT

* *

November 16, 2006

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, ARM Limited, a United Kingdom corporation.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals related to this subject application. There are no interferences related to this subject application.

III. STATUS OF CLAIMS

Claims 1-20 are pending and finally rejected.

IV. STATUS OF AMENDMENTS

No amendment has been filed after final.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Independent claims 1 and 11 are directed to preventing access to sensitive data that should not be accessible by certain data processor applications while still permitting access by one or more other secure applications. An example data processor is a smart card that includes a security application which uses sensitive data like secure keys to perform validation, authentication, decryption and the like. That sensitive data should be kept secure so that it cannot be accessed by other applications that may be loaded onto the smart card data processor, e.g., hacking applications, trying to access that secure data. Page 1, lines 13-21.

Typically, the operating system provides sufficient security to ensure that the secure data of one application cannot be accessed by other applications running under the control of the operating system. But as systems become more complex, a general trend is for operating systems to become larger and more complex, and in such situations, it becomes increasingly difficult to ensure sufficient security within the operating system itself. Claims 1 and 11 describe a new apparatus and technique for protecting the security of secure data stored in a memory. Page 1, lines 22-28.

Figure 1 shows an example useful for understanding the data processing apparatus recited in claims 1 and 11 (these claims are not limited to the example in

Figure 1) that includes a processor core 10 within which is provided an arithmetic logic unit (ALU) 16 arranged to execute sequences of instructions. Data required by the ALU 16 is stored within a register bank 14. The core 10 is coupled to a system bus 40 via memory management logic 30 which is arranged to manage memory access requests issued by the core 10 for access to locations in memory of the data processing apparatus. Certain parts of the memory may be embodied by memory units connected directly to the system bus 40, for example the Tightly Coupled Memory (TCM) 36 and the cache 38 illustrated in Figure 1. Additional devices may also be provided for accessing such memory, for example a Direct Memory Access (DMA) controller 32. Typically, various control registers 34 will be provided for defining certain control parameters of the various elements of the chip, these control registers also being referred to herein as coprocessor 15 (CP 15) registers. The system bus 40 also couples the core 10 and with an external bus 70 to which is coupled an external memory 56, a part of which may be used to store one or more page tables 58 defining information relevant to control of memory accesses. Page 15, lines 8-12; page 16, lines 1-10.

Figure 2 illustrates various application programs (APs) running on the processor in one of a secure (S) domain and a non-secure (NS) domain referred to in claims 1 and 11 (these claims are not limited to the example in Figure 2). A monitor program 72 is responsible for managing the changes between the secure domain and the non-secure domain in either direction. From a view external to the

core, the monitor mode is always secure and the monitor program is in secure memory. Within the non-secure domain, Figure 2 shows a non-secure operating system 74 and a plurality of non-secure application programs 76, 78 which execute in cooperation with the non-secure operating system 74. In the secure domain, a secure kernel program 80 forms a secure operating system executing with secure applications 82 and 84. As a result, when the processor is executing a program in the secure domain, that program has access to secure data which is not accessible from the non-secure domain. Page 17, lines 12-25.

Figure 3 illustrates a matrix of processor modes associated with different domains. In this example, the processing modes are symmetrical with respect to the security domain and accordingly mode 1 and mode 2 exist in both secure and non-secure domains. The monitor mode has the highest level of security access, and in this example, is the only mode entitled to switch the system between the non-secure domain and the secure domain in either direction. Thus, all domain switches take place via a switch to the monitor mode and the execution of the monitor program 72 within the monitor mode. Page 18, lines 1-10.

The memory may include a variety of memory entities, examples of which are shown in Figure 1. Figure 55 illustrates an example, non-limiting embodiment where a mechanism is provided to enable the cache memory 38 and/or the TCM memory 36 to control accesses which are connected via the device bus 70 to one or

more memory units, for example, the external memory 56. Page 98, lines 4-6. One or more of these memories store data at addressable memory entries.

When the core 10 issues a memory access request, this will result in a physical address for that memory access request being output on the system bus 40, and typically, the cache 38 performs a look-up process to determine whether the data item specified by that address is stored within the cache. Whenever a miss occurs within the cache, i.e., it is determined that the data item subject to the access request is not stored within the cache, a linefill procedure is initiated by the cache in order to retrieve from the external memory 56 a line of data that includes the data item that is the subject of the memory access request. In particular, the cache outputs via an external bus interface (EBI) 42 a linefill request onto the control bus 2630 of the device bus 70, with a start address being output on the address bus 2620. In addition, an HPROT signal is output over path 2632 onto the control bus 2630, which includes a domain signal specifying the mode of operation of the core 10 at the time the memory access request was issued. Page 99, lines 14-25.

This HPROT signal identifies to a partition checker 2656, in the example shown in Figure 55, whether the application requesting the specified data from the external memory 56 (in this case the device incorporating the core 10 and the cache 38) was operating in the secure domain or the non-secure domain at the time the memory access request was issued. The partition checker 2656 also has access to the partitioning information identifying which regions of memory are secure or non-

secure, and accordingly, can determine whether the device is allowed to have access to the data it is requesting. Hence, the partition checker 2656 can be arranged to only allow a device to have access to a secure part of the memory if the domain signal within the HPROT signal (also referred to in the detailed description as an S bit) is asserted to identify that access to this data was requested by the device while operating in a secure mode of operation. Page 99, line 29-page 100, line 7.

If the partition checker determines that the core 10 is not allowed to have access to the data requested, for example because the HPROT signal has identified that the core 10 was operating in a non-secure mode of operation, but the linefill request is seeking to retrieve data from the external memory that is within a secure region of memory, then the partition checker 2656 issues an abort signal onto the control bus 2630, which will be passed back over path 2636 to the EBI 42, and from there back to the cache 38, resulting in an abort signal being issued over path 2670 to the core 10. But if the partition checker 2656 determines that the access is allowed, then it outputs an S tag signal identifying whether the data being retrieved from the external memory is secure data or non-secure data. This S Tag signal is passed back via path 2634 to the EBI 42, and from there back to the cache 38 to enable setting of the flag 2602 associated with the cache line 2600 the subject of the linefill process. Page 100, lines 9-20. This flag 2602 is a non-limiting example of the flag recited in claims 1 and 11.

Since, in preferred example embodiments, the original storage of data in a cache line occurs as a result of the above-described linefill process, the flag 2602 associated with that cache line (a cache line is an example of a memory entry) is allocated based on the value provided by the partition checker 2656, and that flag is then used to directly control any subsequent access to the data items in that cache line 2600. Hence, if the core 10 subsequently issues a memory access request that produces a hit in a particular cache line 2600 of the cache 38, the cache 38 will review the value of the associated flag 2602, and compare that value with the current mode of operation of the core 10. In preferred embodiments, this current mode of operation of the core 10 is indicated by a domain bit set by the monitor mode 72 within the CP 15 domain status register. As a result, cache 38 only allows data items in a cache line that the corresponding flag 2602 indicates is secure data to be accessed by the processor core 10 when the processor core 10 is operating in a secure mode of operation. Any attempt by the core 10 to access secure data within the cache 38 while the core 10 is operating in a non-secure mode result in the cache 38 generating an abort signal over path 2670. Page 101, lines 1-15. This is one non-limiting example of how a flag may be used to prevent access to a corresponding data item when the processor is operating in a non-secure mode as recited in claims 1 and 11.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The rejection on appeal is that of claims 1-20 under 35 U.S.C. §102 as being anticipated based upon US-A-2003/0101322 (Gardner).

VII. ARGUMENT

A. The Legal Requirements For Anticipation

To establish that a claim is anticipated, the Examiner must point out where each and every limitation in the claim is found in a single prior art reference. *Scripps Clinic & Research Found. v. Genentec, Inc.*, 927 F.2d 1565 (Fed. Cir. 1991). Every limitation contained in the claims must be present in the reference, and if even one limitation is missing from the reference, then it does not anticipate the claim. *Kloster Speedsteel AB v. Crucible, Inc.*, 793 F.2d 1565 (Fed. Cir. 1986). Gardner is missing several features from the independent claims.

B. Gardner Fails To Teach Multiple Claim Features

1. The Gardner Patent

Gardner describes a mechanism for protecting user application data so that it can be kept secret from root and other users (see Figure 6 and the associated description in paragraphs 0189 to 0193). There are four privilege levels PL0 to PL3: PL0 is the most privileged level of the processor, and PL3 is the least privileged level. The operating system runs at privilege level PL2, and user

applications run at privilege level PL3 (see paragraph 0019). Paragraph 0189 and Figure 6 describe how a user application operating at the least privileged level can keep its data secure from other users. Gardner refers to such user applications that require their data to be kept secure as "secure user applications." To accomplish that security, Gardner uses protection keys which allow a secure user application "to access a page of memory in memory 74 that nobody else can access, including root or anything running at PL2 or above" (see paragraph 0190). The protection keys are inserted into protection key registers by code executing at the protection level PLO (paragraph 0191).

2. Gardner's ELF User Process Security Bit Is Not a Bit Associated With Each Entry in the Memory Unit.

Claim 1 recites a memory unit that includes "a *flag being associated with each entry in the memory unit* to store a value indicating whether the one or more data items stored in the associated entry are said secure data or said non-secure data." A similar feature is found in claim 11. The Examiner reads this claim feature onto Gardner's ELF header, citing paragraph 0189. An ELF header describes the layout of an executable file, and contains information such as the start address of the program code and the type of the code. Hence, each ELF header describes information about a particular piece of program code. As paragraph 0189 explains, secure user *processes* are distinguished from non-secure user *processes* by setting a bit in the ELF header. But distinguishing processes or program code

is not what is claimed. Rather, the claims recite that a flag is associated with *each entry in the memory unit.*

Gardner's ELF header does not associate a flag with each entry in the memory unit. Nor does Gardner disclose some other mechanism for marking individual entries in the memory unit as containing secure data or non-secure data to allow the memory unit to police memory unit accesses. Indeed, Gardner does not describe any details as to how data is stored with a memory unit.

3. Gardner's Memory Unit Does Not Prevent Access From A Non-Secure Domain To Data In An Memory Entry Marked As Storing Secure Data.

Claim 1 further recites: "wherein when the processor is operating in said at least one non-secure mode of the non-secure domain, the memory unit is operable, upon receipt of a memory access request issued by the processor when access to an item of data is required, to prevent access to any data item within an entry of the memory unit that the associated flag indicates has secure data stored therein." A similar feature is found in claim 11. The Examiner points again to Gardner's paragraph 0189: "in one embodiment, the information for distinguishing between secure and non-secure user processes is contained in a secure memory page in memory 74 that cannot be modified by PL2 code." This means that operating system level code cannot tamper with the information that distinguishes between secure and non-secure user *processes*, thereby alleviating the risk that a non-secure

user process can be viewed as being secure via manipulation of that information stored in the secure memory page. But this has nothing to do with the claimed access of data entries in memory.

That Gardner safeguards the information for distinguishing between secure and non-secure user processes so that it cannot be modified by PL2 code is not relevant to claims 1 and 11. Claims 1 and 11 are not concerned with how to manage the issue of a non-secure application or process seeking to disguise itself as a secure application or process. Instead, claims 1 and 11 mark *data* stored in the memory unit on an entry-by-entry basis to identify the data in each memory entry as being either secure data or non-secure data. As a result of that data entry marking, the memory unit prevents access from a non-secure domain to any memory entry marked as storing secure data.

4. Gardner Does Not Perform The Allocation Of Individual Data As Either Secure Or Non-Secure Data In The Secure Domain.

For the claimed memory unit, claims 1 and 11 recite: "the allocation of data as either secure or non-secure data being performed in the secure domain." The Examiner's final action does not address this claimed feature, and it is not found in Gardner. Claims 1 and 11 recite secure and non-secure *domains*, which are *distinct* from the claimed secure and non-secure *modes* of operation.

5. Page Tables 140 And 142 In Gardner Are Not The Claimed Plural Entries In Memory.

Claims 1 and 11 recite: "a memory unit comprising a plurality of entries and operable to store data required by the processor, each entry being operable to store one or more data items including either secure data or non-secure data." The Examiner refers to the page table 140 and the virtual hash page table (VHPT) 142 in Figure 3 of Gardner as the claimed "plurality of entries." The page table 140 is a structure used to map virtual memory pages to physical memory pages. That mapping structure itself is not secure or non-secure data required by the processor (see paragraph 0047). The virtual hash page table (VHPT) is referred to when installing translation entries into the TLB 128 (see paragraph 0057). In contrast, claims 1 and 11 recite that the memory unit entries store secure data or non-secure data required by the processor. Neither the page table 140 nor the virtual hash page table 142 store any such secure or non-secure data.

6. Gardner Fails To Teach The Claimed Secure And Non-Secure Domains.

The Examiner refers to paragraph 0189 of Gardner as teaching secure and non-secure domains. But that paragraph does not mention domains at all. Instead, paragraph 0189 merely refers to secure user applications and non-secure user applications, both of which are user applications running at the lowest privilege level PL3. Claims 1 and 11 employ three different terms: modes, programs, and

domains. The processor can operate in a number of different modes of operation, for example a user mode, a supervisor mode, etc. In a particular mode of operation, the processor can execute programs. For example, a user application or program will typically be executed in a user mode of operation. But in addition to modes of operation and execution of programs, the processor can operate in a plurality of domains including a secure domain and a non-secure domain.

As defined in claims 1 and 11, access to data is performed on a domain basis to ensure that data pertaining to the secure domain is not accessed from the non-secure domain. From Figure 1 in Gardner, the different privilege levels might be equated with different modes of operation. But there is no disclosure of a separate and distinct classification which could reasonably be equated with the additionally claimed secure and non-secure domains.

7. Gardner Fails To Teach The Claimed Cache Memory.

Claims 2 and 12 recite that "the memory unit is a cache, and each said entry is a cache line of the cache." Gardner lacks these claim features. The claimed cache memory unit stores both secure data and non-secure data to be accessed from the secure domain and the non-secure domain, respectively. At the same time, the integrity of the secure data must be protected. Without the approach recited in respective claims 1 and 11, it would be necessary to flush the contents of the cache when transitioning from the secure domain to the non-secure domain to

ensure that there is no inadvertent access to secure data from the non-secure domain.

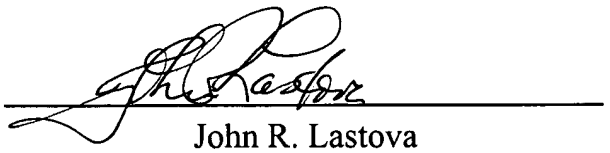
The Examiner refers to paragraph 0137 of Gardner as allegedly teaching the cache recited in claims 2 and 12. But this paragraph merely describes the standard approach of arranging physical memory as a sequence of memory pages. That physical memory clearly is not a cache. Furthermore, Gardner never even addresses the particular problems of maintaining the security of data held in a cache.

VIII. CONCLUSION

There are multiple features from claims 1 and 11 missing from Gardner. The anticipation rejection is in error. The Board should reverse the final rejection and pass this application to allowance.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: 

John R. Lastova
Reg. No. 33,149

JRL/maa
Appendix A - Claims on Appeal

IX. CLAIMS APPENDIX

1. (Previously Presented) A data processing apparatus, comprising:

a processor operable in a plurality of modes and a plurality of domains, said plurality of domains comprising a secure domain and a non-secure domain, said plurality of modes including at least one non-secure mode being a mode in the non-secure domain and at least one secure mode being a mode in the secure domain,

wherein when the processor is in the secure domain, a program executed by the processor has access to secure data which is not accessible from the non-secure domain;

a memory unit comprising a plurality of entries and operable to store data required by the processor, each entry being operable to store one or more data items including either secure data or non-secure data, the allocation of data as either secure or non-secure data being performed in the secure domain, and a flag being associated with each entry in the memory unit to store a value indicating whether the one or more data items stored in the associated entry are said secure data or said non-secure data;

wherein when the processor is operating in said at least one non-secure mode of the non-secure domain, the memory unit is operable, upon receipt of a memory access request issued by the processor when access to an item of data is required, to prevent access to any data item within an entry of the memory unit that the associated flag indicates has secure data stored therein.

2. (Original) A data processing apparatus as claimed in Claim 1, wherein the memory unit is a cache, and each said entry is a cache line of the cache.

3. (Original) A data processing apparatus as claimed in Claim 1, wherein the memory unit is coupled to the processor via a processor bus, the memory unit and processor forming a device, and the data processing apparatus further comprises a device bus via which the device is connectable to a further memory unit, the further memory unit having secure memory for storing secure data and non-secure memory for storing non-secure data.

4. (Original) A data processing apparatus as claimed in Claim 3, wherein if the memory access request specifies a data item that is not stored within the memory unit, the memory access request is output on to the device bus to cause that data item to be accessed in the further memory unit, the data processing apparatus further comprising:

partition checking logic connected to the device bus and operable, whenever the memory access request is issued by the processor when operating in said at least one non-secure mode and is output onto the device bus, to detect if the memory access request is seeking to access the secure memory of the further memory unit, and upon such detection to prevent the access specified by that memory access request.

5. (Original) A data processing apparatus as claimed in Claim 4, wherein if the memory access request specifies a data item that is not stored within the memory unit, then if the partition checking logic determines that the processor is allowed to access that data item, that data item is retrieved from the further memory unit and stored in one of said entries of the memory unit, the value to be set for the flag associated with that entry being indicated by the partition checking logic.

6. (Original) A data processing apparatus as claimed in Claim 3, wherein the further memory unit is a main memory of the data processing apparatus.

7. (Original) A data processing apparatus as claimed in Claim 1, wherein the flag is contained within the memory unit and comprises a single bit set to indicate whether the associated entry has secure data or non-secure data stored therein.

8. (Original) A data processing apparatus as claimed in Claim 1, wherein the memory unit is operable to issue an abort signal if the processor, whilst operating in said at least one non-secure mode, seeks to access any data item within an entry of the memory unit that the associated flag indicates has secure data stored therein.

9. (Original) A data processing apparatus as claimed in Claim 1, wherein the processor is coupled to the memory unit via a memory management unit operable, upon

receipt of the memory access request, to perform one or more predetermined access control functions to control issuance of the memory access request to the memory unit.

10. (Original) A data processing apparatus as claimed in Claim 9, wherein the memory access request specifies a virtual address, and one of said predetermined access control functions comprises conversion of the virtual address to a physical address.

11. (Previously Presented) A method of controlling access to a memory unit of a data processing apparatus, the data processing apparatus comprising a processor operable in a plurality of modes and a plurality of domains, said plurality of domains comprising a secure domain and a non-secure domain, said plurality of modes including at least one non-secure mode being a mode in the non-secure domain and at least one secure mode being a mode in the secure domain, wherein when the processor is in the secure domain, a program executed by said processor has access to secure data which is not accessible from the non-secure domain, the data processing apparatus further comprising a memory unit comprising a plurality of entries and operable to store data required by the processor, each entry being operable to store one or more data items including either secure data or non-secure data, the allocation of data as either secure data or non-secure data being performed in the secure domain, and the method comprising the steps of:

associating a flag with each entry in the memory unit;

when said one or more data items are stored in an entry of the memory unit, storing a value within the associated flag indicating whether said one or more data items are said secure data or said non-secure data;

when the processor is operating in said at least one non-secure mode of the non-secure domain, and upon receipt of a memory access request issued by the processor when access to an item of data is required, preventing access to any data item within an entry of the memory unit that the associated flag indicates has secure data stored therein.

12. (Original) A method as claimed in Claim 11, wherein the memory unit is a cache, and each said entry is a cache line of the cache.

13. (Original) A method as claimed in Claim 11, wherein the memory unit is coupled to the processor via a processor bus, the memory unit and processor forming a device, and the method further comprises the step of:

connecting the device to a further memory unit via a device bus, the further memory unit having secure memory for storing secure data and non-secure memory for storing non-secure data.

14. (Original) A method as claimed in Claim 13, wherein if the memory access request specifies a data item that is not stored within the memory unit, the method further comprises the steps of:

outputting the memory access request on to the device bus to cause that data item to be accessed in the further memory unit;

whenever the memory access request is issued by the processor when operating in said at least one non-secure mode and is output onto the device bus, employing partition checking logic to detect if the memory access request is seeking to access the secure memory of the further memory unit, and upon such detection to prevent the access specified by that memory access request.

15. (Original) A method as claimed in Claim 14, wherein if the memory access request specifies a data item that is not stored within the memory unit, the method further comprises the steps of:

employing the partition checking logic to determine whether the processor is allowed to access that data item; and if so

retrieving that data item from the further memory unit and storing that data item in one of said entries of the memory unit; and

employing the partition checking logic to determine the value to be set for the flag associated with that entry.

16. (Original) A method as claimed in Claim 13, wherein the further memory unit is a main memory of the data processing apparatus.

17. (Original) A method as claimed in Claim 11, wherein the flag is contained within the memory unit and comprises a single bit set to indicate whether the associated entry has secure data or non-secure data stored therein.

18. (Original) A method as claimed in Claim 11, wherein the memory unit is operable to issue an abort signal if the processor, whilst operating in said at least one non-secure mode, seeks to access any data item within an entry of the memory unit that the associated flag indicates has secure data stored therein.

19. (Original) A method as claimed in Claim 11, wherein the processor is coupled to the memory unit via a memory management unit operable, upon receipt of the memory access request, to perform one or more predetermined access control functions to control issuance of the memory access request to the memory unit.

20. (Original) A method as claimed in Claim 19, wherein the memory access request specifies a virtual address, and one of said predetermined access control functions comprises conversion of the virtual address to a physical address.

X. EVIDENCE APPENDIX

There is no evidence appendix.

XI. RELATED PROCEEDINGS APPENDIX

There is no related proceedings appendix.